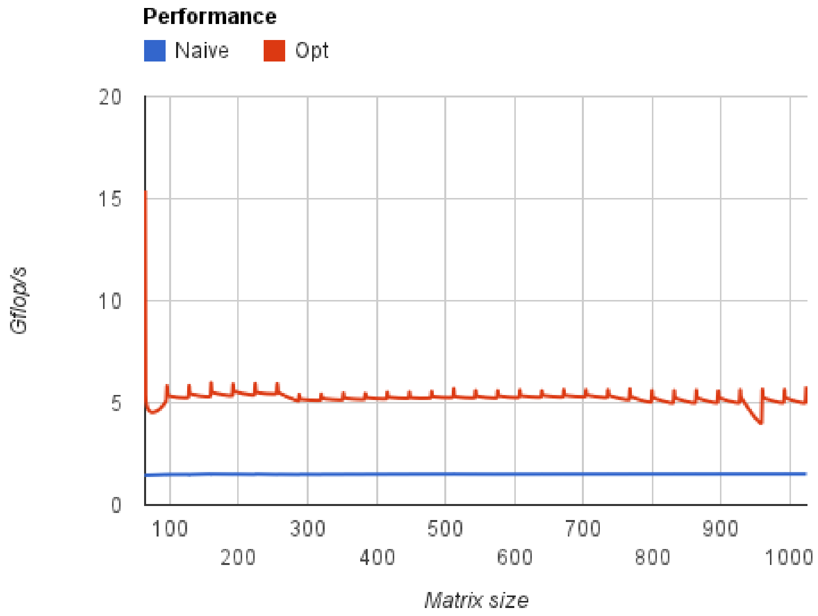
Name: Phuc-Hai Huynh. Login: cs61c-cd
Partner: Khanh Dao. Login: cs61c-ef

1. **Describe how we use SSE registers in the innermost loop of our code**
The order of our loop is j,k,i, so the matrix A and C have a stride of 1, which take advantage of spacial memory, then we can load every four elements of matrix A and C in the column-major order into the SSE registers. But for matrix B, we just able to load one of its element into the SSE registers at a time and fill the rest of register by 0. Then we do the multiplications between SSE registers containing elements of matrices A and B and add them to register which contains elements of matrix C. Since we want to take advantage of all 16 registers, we unroll 8 times in the innermost loop. Also, we use local arrays to contain values of computing in the matrices in the innermost loop to avoid reading from memory many times.

2. **Describe how we deal with the fringes of matrices when N isn't evenly divisible by the SSE register width**
For fringes of matrices when N isn't evenly divisible by the SSE register width, we find the number of times (say chunks) that we can evenly divide the column elements by 4. For instance N = 97, chunks = 97 / 4 = 24 and boundary is 96. From here, using the same method solving for 64x64 matrix, we can calculate all the elements in the matrix C except for the last row (1 row in this case). We calculate these elements by striking B downward direction for N times and A rightward direction for N times. We manually calculate the last row by naive method with 3 for loops. Another example of last rows that we have to calculate naively is N = 102. We have the last 2 rows to calculate naively.

3. **A plot showing the performance of our code as compared to the code in sgemm-navie.c**



4.

The number of XMM registers our code use: 16 registers
Yes. There are value being spilled to the stack during the innermost loop.
The number of scalar floating point instructions:

- There are totally 2 ADDSS and MULSS instructions in assembly code because when the matrix size is not divisible by 4, we have to deal with the fringes of matrices in serial; thus, only one of elements of matrices is loaded into register to do computing.
- There are 36 MOVSS instructions when we loading elements in matrix B.